Overview: This program will be used by registry agent and traffic officers. A registry agent will be able to register a birth, register a marriage, renew a vehicle registration, process a bill of sale, process a payment, and get a driver's abstract. The traffic officer will be able to issue a ticket and find a car owner. A few notes on input, Entered dates must be of the format 'YYYY-MM-DD' where Y,M, and D are single integers. Phone numbers must only contain integers and the '-' character. Registration numbers for birth ,marriages, and vehicles are integers. Ticket numbers are also integers. vehicle's vin are of the form 'UX' where X is an integer. All other inputs have no restrictions.

Software design: Object oriented programming was used to create this program. Main.py starts off by creating a 'session manager' object which creates the login screen and prompts the user for their login information. 'AuthenticationManager.py' is then used to confirm the validity of the user it also provides information about the user. Data about the user can be accessed through 'User.py'. From the information given by ' AuthenticationManager.py ' SessionManager .py' either sends the user to the TrafficOfficer .py' or to the 'RegistryAgent .py' dashboard. The traffic officer and the registry agents uses the following files to implement their function. 'cursor.py' file is used for querying, 'UniqueIDManager.py' is used for making id's unique, SysCallManager .py' is used to switch between different windows or to end the session, 'InputFormatter .py' is used to error check and format any inputs. Each primary function in 'RegistryAgent .py' is modular of other primary functions. User inputs for each question is stored in a dictionary which is mutated accordingly, a list is used to access the dictionary in the proper order. TrafficOfficer.py contains methods for issuing a ticket to registered cars and searching for vehicles based on their make, model, year, color, and license plate. Both the Registry Agent and Traffic Officer inherit from their base class which is User.py.

Testing Strategy: All possible user inputs where tested. The file 'flash.txt' was used to reset the database. it also contained test values that were targeted to testing the functionalities specified in the assignment. Multiple row permutations and duplicates where also addressed by inputting such values in 'flash.txt'.

Group work Breakdown strategy.

The main software architecture was created by bjzukows. bjzukows created the last 3 queries and ipenales created the first 5. This amount of work done bjzukows and ipenales was similar.